Part 1: The Descriptive Set Theory of Finite Structures

(Joint work with Sam Buss, Yijia Chen, Jörg Flum, Moritz Müller)

Standard way to compare NP problems: Polytime reducibility.

However, what if we want to compare isomorphism problems?

$$\mathrm{ISO}(\mathcal{C}) = \{(A, B) \mid A, B \in \mathcal{C} \text{ and } A \simeq B\}$$

Polytime reducibility gives a polytime $f : \mathcal{C} \times \mathcal{C} \to \mathcal{D} \times \mathcal{D}$ such that

$$(A, B) \in \mathrm{ISO}(\mathcal{C}) \leftrightarrow f(A, B) \in \mathrm{ISO}(\mathcal{D})$$

But better would be a polytime $f : \mathcal{C} \to \mathcal{D}$ such that

$$(A, B) \in \mathrm{ISO}(\mathcal{C}) \leftrightarrow (f(A), f(B)) \in \mathrm{ISO}(\mathcal{D})$$

This is called a *strong isomorphism reduction*

Strong Isomorphism Reductions in Complexity Theory

In general: For binary relations E, F on classes $\mathcal{C}, \mathcal{D}$ we define a *strong equivalence reduction* to be a polytime $f : \mathcal{C} \to \mathcal{D}$ such that

$$A \; E \; B \leftrightarrow f(A) \; F \; f(B)$$

This is the finitary analog of the similar theory for *countable* structures, the *Descriptive Set Theory of Equivalence Relations* (big literature)

Summary:

1. Even if $P = \text{NP}$ there is a rich structure to strong isomorphism reducibility.

2. The finer questions about strong isomorphism reducibility are closely connected with deep open problems in complexity theory.

Invariant Polytime Classes

Fix a finite vocabulary (relation, function and constant symbols)

Consider structures for this vocabulary with universe $[n] = \{1, ..., n\}$ for some finite n. (We allow the empty structure.)

Structures can be identified with $0, 1$-strings of length a fixed polynomial in n

The number of structures of size n is exponential in n

A class $\mathcal{C}$ of structures is *invariant* iff it is closed under isomorphism: $A \in \mathcal{C}$, $A \simeq B \rightarrow B \in \mathcal{C}$

$(*)$ We consider polytime, invariant classes with structures of arbitrarily large finite size

Invariant Polytime Classes

Examples:

SET (Empty vocabulary)
FIELD (Fields)
ABELIAN (Abelian groups)
CYCLIC (Cyclic groups)
ORD (Linear orders)
LOP (Linear orders with a distinguished point)
BOOLE (Boolean algebras)
GROUP (Groups)
LOU (Linear orders with a unary relation)
GRAPH (Irreflexive binary relations)

Strong Isomorphism Reductions

If $\mathcal{C}, \mathcal{D}$ are polytime classes then we write: $\mathcal{C} \leq_{iso} \mathcal{D}$ iff

there is a polytime $f : \mathcal{C} \to \mathcal{D}$ such that $A \simeq B \leftrightarrow f(A) \simeq f(B)$

We say that f is a *strong isomorphism reduction* from $\mathcal{C}$ to $\mathcal{D}$

A *strong isomorphism degree* is an equivalence class under the equivalence relation ($\mathcal{C} \leq_{iso} \mathcal{D}$ and $\mathcal{D} \leq_{iso} \mathcal{C}$)

$\mathcal{C} \leq_{iso} \mathcal{D}$ is *stronger* than "ISO($\mathcal{C}$) is polytime reducible to ISO($\mathcal{D}$)".

Indeed, there are many distinct strong isomorphism degrees of classes $\mathcal{C}$ with ISO($\mathcal{C}$) in P

Strong Isomorphism Reductions

Recall that we assume that all of our classes are in P (and therefore all isomorphism relations are in NP)

Proposition

$\mathcal{C} \leq_{iso}$ GRAPH for all $\mathcal{C}$.

Proof Idea: Arbitrary structures can be "coded" as graphs.
Also not difficult:

Proposition

SET, FIELD, ABELIAN, CYCLIC, ORD and LOP all have the same strong isomorphism degree. BOOLE is strong isomorphism reducible to them.

But how do we show that a class $\mathcal{C}$ is NOT strong isomorphism reducible to another class $\mathcal{D}$?

Potential Reducibility

The strong isomorphism complexity of a class is related to the number of isomorphism types it has in different cardinalities

Let $\mathcal{C}$ be a class.
$\mathcal{C}(n) =$ those structures in $\mathcal{C}$ of size at most n
$\#\mathcal{C}(n) =$ number of isomorphism types of structures in $\mathcal{C}(n)$

Examples:

$\#\mathrm{BOOLE}(n) = [logn]$
$\#\mathrm{CYCLIC}(n) = n$, $\#\mathrm{SET}(n) = \#\mathrm{ORD}(n) = n + 1$
$\#\mathrm{LOP}(n) = \sum_{i=1}^{n} i = n \cdot (n+1)/2$
$\#\mathrm{LOU}(n) = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

$\#\mathrm{GROUP}(n)$ is superpolynomial, subexponential
$(\#\mathrm{GROUP}(n) \leq n^{O(log^2 n)})$

Potential Reducibility

For any vocabulary there is a polynomial p such that for any class $\mathcal{C}$ in that vocabulary, $\#\mathcal{C}(n) \leq 2^{p(n)}$

Definition

$\mathcal{C}$ is potentially reducible to $\mathcal{D}$ ($\mathcal{C} \leq_{pot} \mathcal{D}$) iff there is a some polynomial p such that $\#\mathcal{C}(n) \leq \#\mathcal{D}(p(n))$ for all n.

Lemma

If $\mathcal{C} \leq_{iso} \mathcal{D}$ then $\mathcal{C} \leq_{pot} \mathcal{D}$.

Proof. Suppose that f is a strong isomorphism reduction. Then there is a polynomial p such that for A in $\mathcal{C}$, $f(A)$ has size at most $p(|A|)$. As f induces a 1-1 map on isomorphism types, it follows that $\#\mathcal{C}(n) \leq \#\mathcal{D}(p(n))$. $\square$

Potential Reducibility

Now one can show that there is a rich structure of strong isomorphism degrees below LOP:

Theorem

The partial order of the countable atomless Boolean algebra is embeddable into the potential reducibility degrees below LOP. This embedding is also into the strong isomorphism degrees below LOP. In particular, there are infinite chains and antichains in the strong isomorphism degrees below LOP.

SET, FIELD, ABELIAN, CYCLIC, ORD and LOP all have the same potential reducibility degree, as they have the same strong isomorphism degree.

Potential Reducibility

Any class is potentially reducible to LOU, as LOU has the maximum number of isomorphism classes.

Corollary

(a)BOOLE $<_{iso}$ *LOP* $<_{iso}$ *LOU* $\leq_{iso}$ *GRAPH.*
(b) LOP $<_{iso}$ *GROUP* $<_{iso}$ *GRAPH.*
(c) LOU $\not\leq_{iso}$ *GROUP.*

GROUP $\leq_{iso}$ LOU? GRAPH $\leq_{iso}$ LOU?

Potential Reducibility versus Strong Isomorphism Reducibility

Does potential reducibility $\leq_{pot}$ imply strong isomorphism reducibility $\leq_{iso}$? We situate this question between two open questions in complexity theory:

Theorem

If U2EXP $\cap$ co-U2EXP $\neq$ 2EXP then $\leq_{pot}$ and $\leq_{iso}$ are distinct.

2EXP $= \mathrm{DTIME}(2^{2^{n^{O(1)}}})$.

N2EXP $= \mathrm{NTIME}(2^{2^{n^{O(1)}}})$.

U2EXP consists of those problems in N2EXP for which on every input there is exactly one accepting run.

co-U2EXP is the complement of U2EXP.

Potential Reducibility versus Strong Isomorphism Reducibility

Theorem

If $\leq_{pot}$ and $\leq_{iso}$ are distinct then $P \neq \#P$.

For a nondeterministic polytime machine M, define:

$f_M(x) =$ the number of accepting runs of M on input x.

Then the class $\#P$ consists of all such functions f_M.

NP Equivalence Relations

How do isomorphism relations compare to NP equivalence relations as a whole?

For equivalence relations E, F on classes $\mathcal{C}, \mathcal{D}$, a *strong equivalence reduction* is a polytime $f : \mathcal{C} \to \mathcal{D}$ such that

$$A \; E \; B \leftrightarrow f(A) \; F \; f(B)$$

Proposition

If the Polytime Hierarchy does not collapse, then GRAPH is not maximal among NP equivalence relations under strong equivalence reducibility.

The reason is that the maximality of GRAPH would imply that GRAPH is NP-complete in the usual sense, which is known to imply that the Polytime Hierarchy collapses.

Future Work

More facts from Infinite Descriptive Set Theory:

1. Smooth $\rightarrow$ Borel selector for iso relations, not in general.
2. Each closed subgroup of S_∞ is an automorphism group.
3. Iso relations with classes of bounded Borel rank are Borel.
4. Borel ER's with ctble classes are equivalent to iso relations.
5. There is a maximal analytic ER, no maximal Borel ER

Do these have analogues for finite structures?

Part 2: Computational Complexity in Set Theory

Joint work with Arnold Beckmann and Sam Buss

A central notion in Finite computation:

Polytime functions on finite strings

How can we generalise this notion to arbitrary sets?

When is a function $F : V \to V$ computable in "polynomial time"?

Consider some standard models for polynomial-time computation:

Polynomial-Time Set Recursion

1. Turing machines

Difficult to write an arbitrary set on a tape.

2. Fixed point logic

Even for finite structures, this works well only if there is an ordering. Moreover, on infinite ordered structures, it is too powerful (it goes beyond the hyperarithmetic).

3. Schemes

This works, by generalising an idea of Bellantoni-Cook!

Bellantoni-Cook Recursion

Idea behind Bellantoni-Cook recursion:

Define functions

$$f(\vec{x}/\vec{y})$$

where $\vec{x}, \vec{y}$ are finite sequences of finite (binary) strings and the values of f are finite strings

The components of $\vec{x}$ are the *Normal Inputs* and those of $\vec{y}$ the *Safe Inputs*

When performing primitive recursions, the "previous value" is placed on the Safe side:

$f(0, \vec{x}/\vec{y}) = g(\vec{x}/\vec{y})$

$f(z * i, \vec{x}/\vec{y}) = h_i(z, \vec{x}/\vec{y}, f(z, \vec{x}/\vec{y}))$, $i = 0$ or 1

Bellantoni-Cook Recursion

When composing, one is careful not to allow safe inputs to be copied onto the normal side:

$f(\vec{x}/\vec{y}) = h(k(\vec{x}/-)/l(\vec{x}/\vec{y}))$

Net effect: the depth of recursions performed are bounded not by the values obtained but by the sizes of the inputs

On normal inputs one gets exactly the polynomial-time computable functions

On safe inputs string-length is increased by only a constant amount

Bellantoni-Cook Recursion

A few examples (illustrated with numbers, not strings):

The successor function $S(x/y, z) = z + 1$ is in the class

Addition $A(x/y) = x + y$ is in the class, by a primitive recursion:

$A(0/y) = y$
$A(x + 1/y) = S(x/y, A(x/y)) = A(x/y) + 1$

$A^*(x, y/z) = A(y/z) = y + z$ is also in the class

Then multiplication $M(x, y/-) = x \times y$ is in the class, by a second primitive recursion on x:

$M(0, y/-) = 0$
$M(x + 1, y/-) = A^*(x, y/M(x, y/-)) = y + M(x, y/-)$

Bellantoni-Cook Recursion

BUT exponentiation is *not* in the class!

To run the previous argument for exponentiation one would need

$M^*(x, y/z) = M(y/z) = y \times z$

in the class; but we only have

$M(y, z/-)$ (multiplication of Normal Inputs)

and no function

$M(y/z) = y \times z$,

which has z as a Safe Input.

The Safe-Recursive Set Functions

We adapt the Bellantoni-Cook idea to set theory by combining the Gandy-Jensen *rudimentary set functions* with the set-theoretic analogue of Bellantoni-Cook safe recursion

Basic Functions

$\Pi_i^{m,n}(x_1, \ldots, x_m / x_{m+1}, \ldots, x_{m+n}) = x_i \ (1 \leq i \leq m+n)$
$\mathrm{Pair}(-/a, b) = \{a, b\}$
$\mathrm{Diff}(-/a, b) = a \setminus b$

Rudimentary Union Scheme

$f(\vec{x}/\vec{a}, y) = \cup\{g(\vec{x}/\vec{a}, z) \mid z \in y\}$

Safe Recursion

$f(y, \vec{x}/\vec{a}) = h(y, \vec{x}/\vec{a}, \{f(z, \vec{x}/\vec{a}) \mid z \in y\})$

Safe Composition

$f(\vec{x}/\vec{a}) = h(\vec{r}(\vec{x}/-)/\vec{s}(\vec{x}, \vec{a}))$

The Safe-Recursive Set Functions

Some examples of SR set functions:

$S(-/a) = a \cup \{a\}$ is SR

$S^*(a/b, c) = b \cup c$ is SR

$\oplus(a/b) = \{\oplus(c/b) \mid c \in a\} \cup b$ is SR:
$\oplus(a/b) = S^*(a/b, \{\oplus(c/b) \mid c \in a\})$
For ordinals α, β, $\oplus(\alpha/\beta) = \beta + \alpha$

The Safe-Recursive Set Functions

In analogy to getting multiplication from addition through safe-recursion, we have:

$\otimes(a, b/-) = \oplus(b/\{\otimes(c, b) \mid c \in a\})$ is SR

For ordinals α, β, $\otimes(\alpha, \beta/-) = \beta \times \alpha$

But ordinal exponentiation is *not* SR:

Proposition

If $f(\vec{x}/\vec{y})$ is a safe-recursive set function then there is a polynomial function p_f on ordinals such that:

$$rank(f(\vec{x}/\vec{y})) \leq \max(rank(\vec{y})) + p_f(rank(\vec{x}))$$

The Safe-Recursive Set Functions

How powerful are the SR (safe-recursive) functions?

Following Jensen, define:

SR-closure(A) = least SR-closed $B \supseteq A$

For transitive T, SR(T) = SR-closure($T \cup \{T\}$)

We have:

Theorem

For transitive T, $SR(T) = L^T_{rank(T)^\omega}$, where L^T is the L-hierarchy relativised to T.

Characterisation of SRSF Functions

Similarly we have a characterisation of SR functions in terms of definability. For any $\vec{x}$ let $\mathrm{TC}(\vec{x})$ be the transitive closure of $\vec{x}$. The function $\vec{x} \mapsto \mathrm{TC}(\vec{x})$ is SR. Also define:

$\mathrm{SR}(\vec{x}) = \mathrm{SR}(\mathrm{TC}(\vec{x})) = L^{TC(\vec{x})}_{rank(\vec{x})^\omega}$

$\mathrm{SR}_n(\vec{x}) = L^{TC(\vec{x})}_{rank(\vec{x})^n}$ for finite n

Theorem

Suppose that $f(\vec{x}/-)$ is SR. Then for some Σ_1 formula φ and some finite n we have:

$$f(\vec{x}, -) = y \text{ iff } SR_n(\vec{x}) \vDash \varphi(\vec{x}, y).$$

Conversely, any function so defined is SR (on infinite inputs).

The SR Hierarchy

Analog of Jensen's J-hierarchy:

$\mathrm{SR}_1 = \mathrm{HF}$, the collection of hereditarily finite sets
$\mathrm{SR}_{\alpha+1} = \mathrm{SR}(\mathrm{SR}_\alpha)$ for $\alpha > 0$
$\mathrm{SR}_\lambda = \bigcup_{\alpha<\lambda} \mathrm{SR}_\alpha$ for limit λ

Corollary

For every α, $SR_{1+\alpha} = L_{\omega(\omega^\alpha)}$.

$L_\omega \subseteq L_{\omega^\omega} \subseteq L_{\omega(\omega^2)} \subseteq L_{\omega(\omega^3)} \subseteq \cdots$

Safe-Recursion on Restricted Inputs

Binary strings of length ω

If $\vec{x}$ is a finite sequence of binary ω-strings then of course rank$(\vec{x})$ is less than $\omega + \omega$ so we simply have $\mathrm{SR}(\vec{x}) = L_{\omega^\omega}[\vec{x}]$.
Thus the SR functions on ω-strings look like:

$$f(\vec{x}, -) = y \ \textit{iff} \ L_{\omega^n}[\vec{x}] \models \varphi(\vec{x})$$

where φ is Σ_1 and n is finite.

These functions coincide with those computable by an infinite-time Turing machine in time ω^n for some finite n, and were considered by Hamkins, Schindler, Welch and others.

Safe-Recursion on Restricted Inputs

Finite strings
There are many ways to code finite strings as sets

A natural coding:
$\#(i * s) =$ the ordered pair $(i, \#(s)) = \{\{i\}, \{i, \#(s)\}\}$

Theorem

(Beckmann-Buss) With the above coding, the SR functions on finite strings are exactly those in the class $ATIME(EXP, POLY)$ of functions which are computable by an alternating Turing machine running in exponential time with polynomially-many alternations.

Interestingly, the same class arises as the complexity of the first-order theory of the reals with order and addition

Some Recent Work of Tosi Arai

Arai weakened our schemes for SR functions, obtaining his PC (predicatively computable) functions. Recall that we used:

Rudimentary Union Scheme:

$f(\vec{x}/\vec{a}, y) = \cup\{g(\vec{x}/\vec{a}, z) \mid z \in y\}$

Arai weakens this to:

$f(\vec{x}, y/\vec{a},) = \cup\{g(\vec{x}, z/\vec{a}) \mid z \in y\}$

Theorem

(Arai) The PC functions on finite strings are exactly the PTIME functions.

On infinite inputs, SR = PC

Future Work

1. Another strategy to define complexity classes in Set Theory: Definability in Weak Set Theories.

There is work of Arai, Buss and others on this for PTIME.

2. Further machine models for complexity classes in set theory (wellordered inputs).

3. Descriptive complexity in set theory: logics that capture complexity classes?